CLAIMS

What is claimed is:

1. A method for dispatching instructions executed by at least one functional unit of a data processor, each one of the instructions having a corresponding priority number, in a computer system having at least one host processor and host memory, the method comprising:

receiving a next instruction;

examining a current instruction group to determine if the current instruction group is completed;

adding the next instruction to the current instruction group if the current instruction group is not completed; and

dispatching the current instruction group if the current instruction group is completed.

2. The method of claim 1, wherein if the current instruction group is completed, the method further comprises:

starting a new instruction group; and

adding the next instruction to the new instruction group.

3. The method of claim 1, further comprising:

examining the next instruction to determine if the corresponding priority number of the next instruction is equal to or lower than the corresponding priority number of a current instruction of the current instruction group;

adding the next instruction to the current instruction group if the corresponding priority number of the next instruction is higher than the corresponding priority number of the current instruction of the current instruction group; and

dispatching the current instruction group if the corresponding priority number of the next instruction is equal to or lower than the corresponding priority number of the current instruction of the current instruction group.

4. The method of claim 3, wherein if the corresponding priority number of the next instruction is higher than the corresponding priority number of the current instruction of the current instruction group, the method further comprises:

examining the next instruction to determine if the next instruction is required to be in a new instruction group;

wherein if the next instruction is required to be in a new instruction group:

adding a no-operation (NOOP) instruction to the current instruction group;

dispatching the current instruction group;

starting a new instruction group; and

adding the next instruction to the new instruction group.

5. The method of claim 3, wherein if the corresponding priority number of the next instruction is higher than the corresponding priority number of the current instruction of the current instruction group, the method further comprises:

examining the current instruction group to determine if the current instruction group contains a predetermined number of instructions;

wherein if the current instruction group contains the predetermined number of instructions:

dispatching the current instruction group;

starting a new instruction group; and

adding the next instruction to the new instruction group.

6. The method of claim 1, further comprising:

examining the current instruction group to determine if the current instruction group contains a predetermined number of instructions; and

dispatching the current instruction group if the current instruction group contains the predetermined number of instructions.

7. The method of claim 1, wherein all instructions in the current instruction group are dispatched in the same clock cycle.

8. The method of claim 1, further comprising:

examining the next instruction to determine latency required by the next instruction;

calculating delay cycles based on the latency; and

suspending the dispatching for a period of time corresponding to the delay cycles.

9. The method of claim 8, further comprising inserting an additional delay cycle during the suspension.

10. The method of claim 1, further comprising:

examining the next instruction to determine if the next instruction contains an illegal operation code; and

issuing an error message through an interrupt mechanism, if the next instruction contains an illegal operation code.

11. The method of claim 1, wherein if the next instruction is a non-branch instruction, the method further comprises:

examining the next instruction to determine if source resources required by the next instruction are in-use; and

stalling instruction dispatching if the source resources required by the next instruction are in-use.

12. The method of claim 11, wherein the source resources are defined by source operand registers required by the next instruction.

13. The method of claim 1, wherein if the next instruction is a non-branch instruction, the method further comprises:

examining the next instruction to determine if destination resources required by the next instruction are in-use; and

stalling instruction dispatching if the destination resources required by the next instruction are in-use.

14. The method of claim 13, wherein the destination resources are defined by target destination registers required by the next instruction.

15. The method of claim 1, wherein if the next instruction is a branch instruction, the method further comprises:

examining resources required by the branch instruction to determine if the resources are used or altered by a non-branch instruction; and

wherein if the resources are used or altered by a non-branch instruction, suspending the dispatching the next instruction until the resources are available.

16. The method of claim 15, further comprising inserting an additional delay cycle during the suspension.

17. The method of claim 5, wherein the predetermined number of instructions comprises four instructions.

18. The method of claim 6, wherein the predetermined number of instructions comprises four instructions.

19. The method of claim 3, further comprising accessing a database to determine the corresponding priority number of the next instruction.

20. The method of claim 8, further comprising accessing a database to determine the latency required by the next instruction.

21. The method of claim 1, wherein the data processor is integrated in a system core logic chip that functions as a bridge between the host processor and the host memory, and other components of the computer system, the system core logic chip having a host interface coupled to the host processor and a memory interface coupled to the host memory.

22. The method of claim 1, wherein the data processor may be a stand-alone processor, or the data processor may be a co-processor to the host processor.

23. The method of claim 1, wherein the at least one functional unit comprises multiple functional units of a kind.

24. The method of claim 23, further comprising:

examining the next instruction to determine if there is a corresponding functional unit that executes the next instruction available;

adding the next instruction to the current instruction group if the corresponding functional unit is available; and

dispatching the current instruction group if the corresponding functional unit is not available.

25. The method of claim 24, wherein if the corresponding functional unit that executes the next instruction is available, the method further comprises:

examining the next instruction to determine if the next instruction is required to be in a new instruction group;

wherein if the next instruction is required to be in a new instruction group:

adding a no-operation (NOOP) instruction to the current instruction group;

dispatching the current instruction group;

starting a new instruction group; and

adding the next instruction to the new instruction group.

26. The method of claim 24, wherein if the corresponding functional unit that executes the next instruction is available, the method further comprises:

examining the current instruction group to determine if the current instruction group contains a predetermined number of instructions;

wherein if the current instruction group contains the predetermined number of instructions:

dispatching the current instruction group;

starting a new instruction group; and

adding the next instruction to the new instruction group.

27. The method of claim 26, wherein the predetermined number of instructions comprises four instructions.

28. An apparatus for dispatching instructions executed by at least one functional unit of a data processor, each one of the instructions having a corresponding priority number, in a computer system having at least one host processor and host memory, the apparatus comprising:

means for receiving a next instruction;

means for examining a current instruction group to determine if the current instruction group is completed;

means for adding the next instruction to the current instruction group if the current instruction group is not completed; and

means for dispatching the current instruction group if the current instruction group is completed.

29. The apparatus of claim 28, wherein if the current instruction group is completed, the apparatus further comprises:

means for starting a new instruction group; and

means for adding the next instruction to the new instruction group.

30. The apparatus of claim 28, further comprising:

means for examining the next instruction to determine if the corresponding priority number of the next instruction is equal to or lower than the corresponding priority number of a current instruction of the current instruction group;

means for adding the next instruction to the current instruction group if the corresponding priority number of the next instruction is higher than the corresponding priority number of the current instruction of the current instruction group; and

means for dispatching the current instruction group if the corresponding priority number of the next instruction is equal to or lower than the corresponding priority number of the current instruction of the current instruction group.

31. The apparatus of claim 30, wherein if the corresponding priority number of the next instruction is higher than the corresponding priority number of the current instruction of the current instruction group, the apparatus further comprises:

means for examining the next instruction to determine if the next instruction is required to be in a new instruction group;

wherein if the next instruction is required to be in a new instruction group:

means for adding a no-operation (NOOP) instruction to the current instruction group;

means for dispatching the current instruction group;

means for starting a new instruction group; and

means for adding the next instruction to the new instruction group.

32. The apparatus of claim 30, wherein if the corresponding priority number of the next instruction is higher than the corresponding priority number of the current instruction of the current instruction group, the apparatus further comprises:

means for examining the current instruction group to determine if the current instruction group contains a predetermined number of instructions;

wherein if the current instruction group contains the predetermined number of instructions:

means for dispatching the current instruction group;

means for starting a new instruction group; and

means for adding the next instruction to the new instruction group.

33. The apparatus of claim 28, further comprising:

means for examining the current instruction group to determine if the current instruction group contains a predetermined number of instructions; and

means for dispatching the current instruction group if the current instruction group contains the predetermined number of instructions.

34. The apparatus of claim 28, wherein all instructions in the current instruction group are dispatched in the same clock cycle.

35. The apparatus of claim 28, further comprising:

means for examining the next instruction to determine latency required by the next instruction;

means for calculating delay cycles based on the latency; and

means for suspending the dispatching for a period of time corresponding to the delay cycles.

36. The apparatus of claim 35, further comprising means for inserting an additional delay cycle during the suspension.

37. The apparatus of claim 28, further comprising:

means for examining the next instruction to determine if the next instruction contains an illegal operation code; and

means for issuing an error message through an interrupt mechanism, if the next instruction contains an illegal operation code.

38. The apparatus of claim 28, wherein if the next instruction is a non-branch instruction, the apparatus further comprises:

means for examining the next instruction to determine if source resources required by the next instruction are in-use; and

means for dispatching a no-operation (NOOP) instruction if the source resources required by the next instruction are in-use.

39. The apparatus of claim 38, wherein the source resources are defined by source operand registers required by the next instruction.

40. The apparatus of claim 28, wherein if the next instruction is a non-branch instruction, the apparatus further comprises:

means for examining the next instruction to determine if destination resources required by the next instruction are in-use; and

means for dispatching a no-operation (NOOP) instruction if the destination resources required by the next instruction are in-use.

41. The apparatus of claim 40, wherein the destination resources are defined by target destination registers required by the next instruction.

42. The apparatus of claim 28, wherein if the next instruction is a branch instruction, the apparatus further comprises:

means for examining resources required by the branch instruction to determine if the resources are used or altered by a non-branch instruction; and

means for wherein if the resources are used or altered by a non-branch instruction, suspending the dispatching the next instruction until the resources are available.

43. The apparatus of claim 42, further comprising means for inserting an additional delay cycle during the suspension.

44. The apparatus of claim 32, wherein the predetermined number of instructions comprises four instructions.

45. The apparatus of claim 33, wherein the predetermined number of instructions comprises four instructions.

46. The apparatus of claim 30, further comprising means for accessing a database to determine the corresponding priority number of the next instruction.

47. The apparatus of claim 35, further comprising means for accessing a database to determine the latency required by the next instruction.

48. The apparatus of claim 28, wherein the data processor is integrated in a system core logic chip that functions as a bridge between the host processor and the host memory, and other components of the computer system, the system core logic chip having a host interface coupled to the host processor and a memory interface coupled to the host memory.

49. The apparatus of claim 28, wherein the data processor may be a stand-alone processor, or the data processor may be a co-processor to the host processor.

50. The apparatus of claim 28, wherein the at least one functional unit comprises multiple functional units of a kind.

51. The apparatus of claim 50, further comprising:

means for examining the next instruction to determine if there is a corresponding functional unit that executes the next instruction available;

means for adding the next instruction to the current instruction group if the corresponding functional unit is available; and

means for dispatching the current instruction group if the corresponding functional unit is not available.

52. The apparatus of claim 51, wherein if the corresponding functional unit that executes the next instruction is available, the apparatus further comprises:

means for examining the next instruction to determine if the next instruction is required to be in a new instruction group;

wherein if the next instruction is required to be in a new instruction group:

means for adding a no-operation (NOOP) instruction to the current instruction group;

means for dispatching the current instruction group;

means for starting a new instruction group; and

means for adding the next instruction to the new instruction group.

53. The apparatus of claim 51, wherein if the corresponding functional unit that executes the next instruction is available, the apparatus further comprises:

means for examining the current instruction group to determine if the current instruction group contains a predetermined number of instructions;

wherein if the current instruction group contains the predetermined number of instructions:

means for dispatching the current instruction group;

means for starting a new instruction group; and

means for adding the next instruction to the new instruction group.

54. The apparatus of claim 53, wherein the predetermined number of instructions comprises four instructions.

55. A machine readable medium having stored thereon executable code which causes a machine to perform a method, for dispatching instructions executed by at least one functional unit of a data processor, each one of the instructions having a corresponding priority number, in a computer system having at least one host processor and host memory, the method comprising:

receiving a next instruction;

examining a current instruction group to determine if the current instruction group is completed;

adding the next instruction to the current instruction group if the current instruction group is not completed; and

dispatching the current instruction group if the current instruction group is completed.

56. The machine readable medium of claim 55, wherein if the current instruction group is completed, the method further comprises:

starting a new instruction group; and

adding the next instruction to the new instruction group.

57. The machine readable medium of claim 55, wherein the method further comprises:

examining the next instruction to determine if the corresponding priority number of the next instruction is equal to or lower than the corresponding priority number of a current instruction of the current instruction group;

adding the next instruction to the current instruction group if the corresponding priority number of the next instruction is higher than the corresponding priority number of the current instruction of the current instruction group; and

dispatching the current instruction group if the corresponding priority number of the next instruction is equal to or lower than the corresponding priority number of the current instruction of the current instruction group.

58. The machine readable medium of claim 57, wherein if the corresponding priority number of the next instruction is higher than the corresponding priority number of the current instruction of the current instruction group, the method further comprises:

examining the next instruction to determine if the next instruction is required to be in a new instruction group;

wherein if the next instruction is required to be in a new instruction group:

adding a no-operation (NOOP) instruction to the current instruction group;

dispatching the current instruction group;

starting a new instruction group; and

adding the next instruction to the new instruction group.

59. The machine readable medium of claim 57, wherein if the corresponding priority number of the next instruction is higher than the corresponding priority number of the current instruction of the current instruction group, the method further comprises:

examining the current instruction group to determine if the current instruction group contains a predetermined number of instructions;

wherein if the current instruction group contains the predetermined number of instructions:

dispatching the current instruction group;

starting a new instruction group; and

adding the next instruction to the new instruction group.

60. The machine readable medium of claim 55, wherein the method further comprises:

examining the current instruction group to determine if the current instruction group contains a predetermined number of instructions; and

dispatching the current instruction group if the current instruction group contains the predetermined number of instructions.

61. The machine readable medium of claim 55, wherein all instructions in the current instruction group are dispatched in the same clock cycle.

62. The machine readable medium of claim 55, wherein the method further comprises:

examining the next instruction to determine latency required by the next instruction;

calculating delay cycles based on the latency; and

suspending the dispatching for a period of time corresponding to the delay cycles.

63. The machine readable medium of claim 62, wherein the method further comprises inserting an additional delay cycle during the suspension.

64. The machine readable medium of claim 55, wherein the method further comprises:

examining the next instruction to determine if the next instruction contains an illegal operation code; and

issuing an error message through an interrupt mechanism, if the next instruction contains an illegal operation code.

65. The machine readable medium of claim 55, wherein if the next instruction is a non-branch instruction, the method further comprises:

examining the next instruction to determine if source resources required by the next instruction are in-use; and

dispatching a no-operation (NOOP) instruction if the source resources required by the next instruction are in-use.

66. The machine readable medium of claim 65, wherein the source resources are defined by source operand registers required by the next instruction.

67. The machine readable medium of claim 55, wherein if the next instruction is a non-branch instruction, the method further comprises:

examining the next instruction to determine if destination resources required by the next instruction are in-use; and

dispatching a no-operation (NOOP) instruction if the destination resources required by the next instruction are in-use.

68. The machine readable medium of claim 67, wherein the destination resources are defined by target destination registers required by the next instruction.

69. The machine readable medium of claim 55, wherein if the next instruction is a branch instruction, the method further comprises:

examining resources required by the branch instruction to determine if the resources are used or altered by a non-branch instruction; and

wherein if the resources are used or altered by a non-branch instruction, suspending the dispatching the next instruction until the resources are available.

70. The machine readable medium of claim 69, wherein the method further comprises inserting an additional delay cycle during the suspension.

71. The machine readable medium of claim 59, wherein the predetermined number of instructions comprises four instructions.

72. The machine readable medium of claim 60, wherein the predetermined number of instructions comprises four instructions.

73. The machine readable medium of claim 57, wherein the method further comprises accessing a database to determine the corresponding priority number of the next instruction.

74. The machine readable medium of claim 62, wherein the method further comprises accessing a database to determine the latency required by the next instruction.

75. The machine readable medium of claim 55, wherein the data processor is integrated in a system core logic chip that functions as a bridge between the host processor and the host memory, and other components of the computer system, the system core logic chip having a host interface coupled to the host processor and a memory interface coupled to the host memory.

76. The machine readable medium of claim 55, wherein the data processor may be a stand-alone processor, or the data processor may be a co-processor to the host processor.

77. The machine readable medium of claim 55, wherein the at least one functional unit comprises multiple functional units of a kind.

78. The machine readable medium of claim 77, wherein the method further comprises:

examining the next instruction to determine if there is a corresponding functional unit that executes the next instruction available;

adding the next instruction to the current instruction group if the corresponding functional unit is available; and

dispatching the current instruction group if the corresponding functional unit is not available.

79. The machine readable medium of claim 78, wherein if the corresponding functional unit that executes the next instruction is available, the method further comprises:

examining the next instruction to determine if the next instruction is required to be in a new instruction group;

wherein if the next instruction is required to be in a new instruction group:

adding a no-operation (NOOP) instruction to the current instruction group;

dispatching the current instruction group;

starting a new instruction group; and

adding the next instruction to the new instruction group.

80. The machine readable medium of claim 78, wherein if the corresponding functional unit that executes the next instruction is available, the method further comprises:

examining the current instruction group to determine if the current instruction group contains a predetermined number of instructions;

wherein if the current instruction group contains the predetermined number of instructions:

dispatching the current instruction group;

starting a new instruction group; and

adding the next instruction to the new instruction group.

81. The machine readable medium of claim 80, wherein the predetermined number of instructions comprises four instructions.

82. An apparatus for dispatching instructions executed by at least one functional unit of a data processor, the apparatus comprising:

an instruction cache memory for receiving instructions from an input and output (I/O) interface;

an instruction decoder coupled to construct an instruction group based on the priorities of the instructions; and

a dispatch controller coupled to dispatch the instruction group to an appropriate functional unit.

83. The apparatus of claim 82, further comprising:

at least one instruction registers coupled to store the instructions being grouped; and

at least one instruction buffers coupled to store instructions when the instruction fetching is stalled.

84. The apparatus of claim 82, further comprising a branch decoder coupled to detect a branch condition and to generate the address for the next instruction being fetched.

85. The apparatus of claim 84, further comprising a program counter coupled to receive commands from the branch decoder to fetch the next instruction at the address.

86. The apparatus of claim 83, wherein the instruction decoder retrieves the instructions from the at least one instruction registers or from the at least one instruction buffers after the instruction stalling cycles.

87. The apparatus of claim 82, wherein the instruction decoder stalls the instruction fetching based on the latency of the instruction being executed.